Interoperabilidad de Componentes Software mediante Lenguajes de Coordinación [*]

Silvia Amaro, Nadina Martinez Carod
Departamento de Informática y Estadística, Universidad Nacional del Comahue,
Buenos Aires 1400, Neuquén, Argentina
TE: (0299) 4490312 - Fax:(0299) 4490313
Email: samaro, namartin@uncoma.edu.ar

Ernesto Pimentel
Departammento de Lenguajes y Ciencias de la Computación, Universidad de Málaga,
Campus Universitario, Málaga, España
Email: ernesto@lcc.uma.es

Resumen El desarrollo de software basado en componentes es una disciplina en continuo crecimiento dentro del campo de la Ingeniería de software. Con las primeras plataformas de componentes del mercado surgió el desarrollo de los lenguajes de descripción de interfaces con la idea de detectar problemas de compatibilidad entre componentes en el desarrollo de aplicaciones distribuidas y abiertas. Sin embargo, al describir los servicios que ofrecen los objetos, los lenguajes de descripción de interfaces sólo consideran la signatura de los métodos, obviando el problema principal de la composición de componentes respecto al protocolo de interacción que les permite comunicarse y cooperar. Nuestro trabajo actual se enfoca en el análisis del uso de Manifold, un modelo de coordinación orientado a control, como base para un formalismo en cuyo contexto definiremos relaciones que permitan analizar las propiedades para que el comportamiento interactivo de componentes software sea seguro.

1. Introducción

Para responder a la creciente demanda de nuevas tecnologías para la construcción de aplicaciones abiertas y distribuidas, la Ingeniería de Software Basada en Componentes (CBSE) propone la creación de colecciones de componentes de software reutilizables que puedan ser adaptadas e interconectadas en forma dinámica en el desarrollo de nuevas aplicaciones. El reuso de componentes permite adaptar aplicaciones cambiando componentes existentes o introduciendo nuevas. Para que la búsqueda y recuperación de componentes para su ensamblado en la creación de nuevas aplicaciones tenga éxito es necesario verificar que el comportamiento de las componentes es el apropiado para que puedan interoperar, es decir "comunicarse y cooperar a pesar de las diferencias en los lenguajes de implementación, el ambiente de ejecución o la abstracción del modelo" [15].

Las plataformas de componentes actuales, como CORBA, EJB, COM, .NET facilitan el desarrollo de aplicaciones concurrentes y abiertas. Mediante los lenguajes de descripción de interfaces (IDL) que proveen, se generan descripciones estáticas de los servicios ofrecidos por las componentes. Sin embargo, los únicos mecanismos que pueden ofrecer para determinar la conveniencia o no de integrar una componente ya existente en un contexto determinado es la comprobación de que el nombre de los servicios requeridos por parte del sistema coincide con los que son proporcionados por la componente localizada, pero nada se puede decir de la forma en que esos servicios deben ser utilizados.

Actualmente existen varias propuestas de extensiones a los IDL tradicionales. Se han utilizado álgebras de procesos para describir los protocolos de interacción y se han definido formalismos que establecen relaciones para medir el nivel de compatibilidad entre componentes, y como consecuencia la posibilidad de reemplazar unas componentes por otras [7, 10, 13]. Otros trabajos [11] dan una vista

[*] Este trabajo es financiado parcialmente por CYTED (Ciencia y Tecnología para el desarrollo), proyecto VII_J_RITOS2 (Red Iberoamericana de Tecnologías de Software para la década del 2000)

mas práctica presentando la aplicación de álgebras de procesos a plataformas de componentes ampliamente difundidas. La próxima sección introduce brevemente los modelos de coordinación y sus posibilidades en la interoperabilidad de componentes. Finalmente se dan algunas conclusiones

2. Modelos de coordinación para la interacción de componentes software

Uno de los principales objetivos a lograr al trabajar con componentes software heterogéneas es conseguir una separación clara entre los aspectos de interacción y de computación, para así favorecer la reusabilidad de componentes, facilitando el análisis global de la aplicación [2]. En este sentido, la alternativa ofrecida por los lenguajes y modelos de coordinación (Linda [12] y Manifold [1], por ejemplo) resulta prometedora.

2.1 Lenguaje de coordinación Linda

Linda es un modelo de coordinación en el que el sistema está formado por agentes que desarrollan sus actividades de forma independiente y cuya comunicación se logra de alguna manera asincrónica a través de un medio compartido, referido como espacio de tuplas. Su modelo de comunicación posee una gran potencia expresiva para especificar sistemas distribuidos y concurrentes [9].

Actualmente se está explorando la aplicación de Linda en el terreno de la interoperabilidad de componentes software. En [8] se propone Linda para describir el comportamiento interactivo de componentes software, junto con una relación de compatibilidad -definida a partir de un álgebra de procesos basado en el modelo- que permite establecer comprobaciones dinámicas de la compatibilidad de interfaces. Dado que la evolución del espacio de tuplas gobierna cada paso del cómputo basado en Linda, la referencia al entorno en la relación de compatibilidad definida es escencial, ya que proporciona información relevante respecto al estado del sistema, representado por el conjunto de tuplas incluidas en el espacio de datos. La composición segura de componentes sensible al entorno permite comprobar la compatibilidad de un componente y un sistema en ejecución con respecto al entorno actual, y puede condicionar la incorporación de una componente dada a un sistema en ejecución hasta que el espacio de tuplas alcance un estado adecuado con el objeto de asegurar el éxito del sistema globalmente.

2.2 Manifold

Manifold es un lenguaje de coordinación orientado a control en el cual la comunicación entre procesos se realiza por medio de conexiones de flujos (*streams*) punto a punto. Los procesos que intervienen en un patrón de coordinación se definen por medio de un estado formado por el conjunto de conexiones de flujos observables y reaccionan ante la presencia de eventos [1]. Su modelo conceptual se basa en el modelo IWIM [3], según el cual existen procesos que juegan el rol de *workers*, y procesos *manager*, coordinadores en el caso de Manifold.

Posee características interesantes -composicionalidad, comunicación anónima, evolución de la coordinación en reacción a eventos, separación de coordinación y computación- que lo disponen como un lenguaje apropiado para desarrollar configuraciones de componentes que evolucionan dinámicamente [4, 14].

Las capacidades del lenguaje no han sido suficientemente exploradas aún. Nuestro primer objetivo es utilizar Manifold para describir el protocolo de componentes y proponer una extensión que permita analizar distintas componentes y determinar si su composición paralela estará libre de bloqueos. Para ello se debe definir una relación de compatibilidad que sea decidible, ya que nuestro segundo objetivo es implementar una herramienta que compruebe automáticamente dicha compatibilidad. Por último nos interesa además, por medio de alguna relación de sustitutabilidad decidible, proponer condiciones

suficientes para garantizar, a partir de la descripción del protocolo de una componente, que dicha componente podrá sustituir a otra existente, manteniendo la compatibilidad con el resto del sistema.

Conclusiones y trabajo futuro

Los distintos lenguajes de coordinación comparten características interesantes, tales como desacople de tiempo y espacio, y una gran potencia expresiva para especificar sistemas distribuidos y concurrentes. Motivados por ello, una vez logrado nuestro objetivo respecto a la interoperabilidad de componentes utilizando Manifold, nuestro trabajo estará orientado a obtener un resultado aplicable a los lenguajes de coordinación en general. Para lograrlo estamos planeando combinar los resultados obtenidos con nuestro trabajo en Manifold, los resultados presentados en [8] respecto a la interoperabilidad de componentes en Linda, y las relaciones entre los distintos estilos arquitectónicos y las diversas familias de lenguajes de coordinación presentados en [5, 6].

Referencias

1. F. Arbab, *Manifold Version 2.0*. 2000
2. F. Arbab, *What do you mean, coordination?*, Bulletin of the Dutch Association for theoretical Computer Science, NVTI, pages 11-22, 1998.
3. F. Arbad, *The IWIM Model for Coordination of Concurrent Activities*, Proceedings First International Conference on Coordination Models, Languages and Applications (Coordination'96), pp 34-56.
4. F. Arbab, M.M. Bonsangue, F.S. de Boer, *A coordination Language for Mobile Components*, Proceedings of SAC 2000, ACM Press, pages 166-173, 2000.
5. M.M. Bonsangue, J.N. Kok, G. Zavattaro, *Comparing Software Architectures for Coordination Languages*, Proceedings of Coordination 99, Vol 1594 of Lecture Notes in Computer Science, pages 150-165, 1999.
6. M.M. Bonsangue, J.N. Kok, G. Zavattaro, *Comparing Coordination Models and Architectures using embeddings*, Reporte SEN-R0025, Centrum voor Wiskunde en Informatica (CWI), 2000
7. A. Bracciali, A. Brogi, F. Turini, *Interaction patterns*, V Jornadas Iberoamericanas de Ingeniería de Software, 2001.
8. A. Brogi, E. Pimentel, A.M. Roldán, *Interoperabilidad de Componentes Software en Linda*, IDEAS 2002
9. A. Brogi, J.M. Jacquet, *On the expressiveness of coordination models*, Coordination Languages and Models:3rd. International Conference, Vol. 1594 of Lecture Notes, pages 134-149,1999.
10. C. Canal, *Un lenguaje para la especificación y validación de arquitecturas de software*, Ph.D. thesis, Depto Lenguajes y Ciencias de la Computación, Universidad de Málaga, 2001.
11. C. Canal, L. Fuentes, J.M. Troya, and A. Vallecillo, *Extending CORBA interfaces with pi-calculus for protocol compatibility*, Proceedings of the Technology of Object-Oriented Languages and Systems - TOOLS Europe 2000, IEEE Press 2000, pp. 208-225
12. D. Gelernter and N. Carriero, *Coordination Languages and their significance*, Communications of the ACM, No. 35, pp. 97-107. 1992
13. G.T. Leavens, M. Staraman, *Foundations of component based systems*, Cambridge University Press, 2000.
14. G. Papadopoulos and F. Arbab, *Dynamic Reconfiguration in Coordination Languages*, Advances in Computers 46, Marvin V. Zelkowitz, Academic Press, 1998, pp. 329-400
15. P. Wegner, *Interoperability*, ACM Computing Surveys, Vol. 28, No. 1, March 1996.